ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 222E

COMPUTER ORGANIZATION PROJECT REPORT

PROJECT NO : 3

PROJECT DATE : 03.06.2020

GROUP NO : G22

GROUP MEMBERS:

150180903 : Khayal HUSEYNOV

150180901 : Ramal SEYIDLI

150180725 : İsmayil Buğra KÜTÜKOĞLU

150180720 : Kerim GENÇ

Contents

1 INTRODUCTION

In project 3, we designed a Basic Computer and it is look like shown below. We made operations possible by using project 2. While we have designed Basic Computer in project 2 indeed, we updated it to control unit in this project.

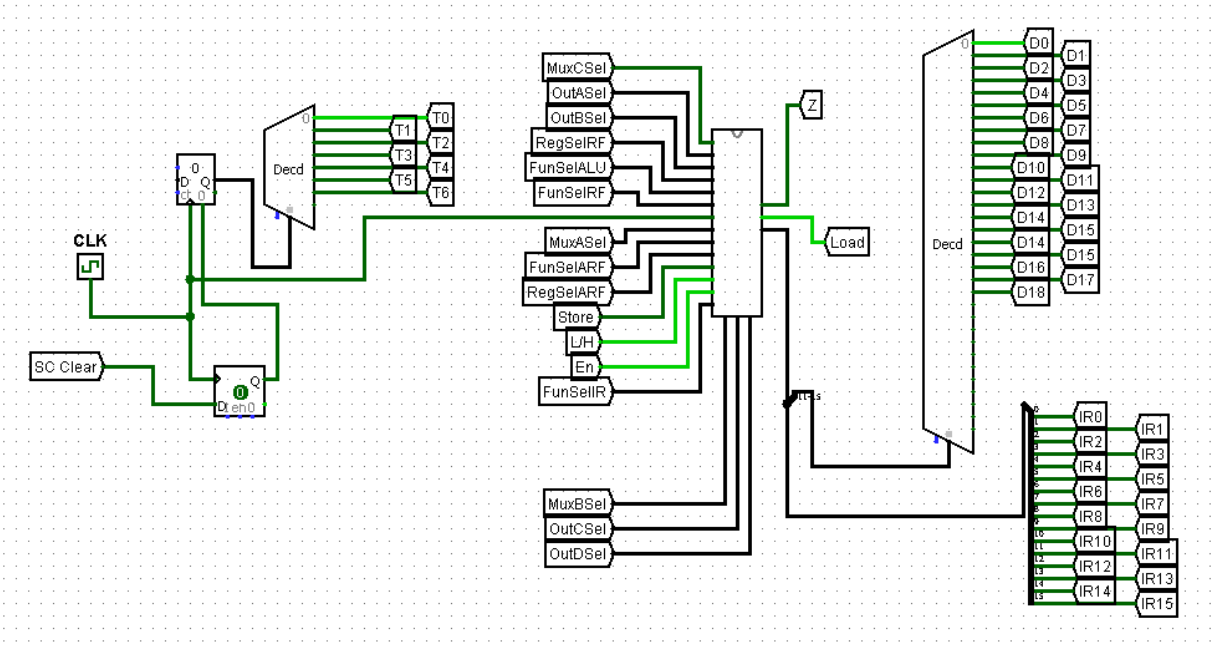

Figure 1: Main Circuit

We made it possible by applying instructions of opcode below:

OPCODE (HEX)	SYMB	ADDRESSING MODE	DESCRIPTION
0x00	LD	IM, D	Rx ← Value (Value is described in Table 3)
0x01	ST	D	Value ← Rx
0x02	MOV	N/A	DESTREG ← SRCREG1
0x03	PSH	N/A	M[SP] ← Rx, SP ← SP - 1
0x04	PUL	N/A	SP ← SP + 1, Rx ← M[SP]
0x05	ADD	N/A	DESTREG ← SRCREG1 + SRCREG2
0x06	SUB	N/A	DESTREG ← SRCREG2 - SRCREG1
0x07	DEC	N/A	DESTREG ← SRCREG1 - 1
0x08	INC	N/A	DESTREG ← SRCREG1 + 1
0x09	AND	N/A	DESTREG ← SRCREG1 AND SRCREG2
0x0A	OR	N/A	DESTREG ← SRCREG1 OR SRCREG2
0x0B	NOT	N/A	DESTREG ← NOT SRCREG1
0x0C	LSL	N/A	DESTREG ← LSL SRCREG1
0xOD	LSR	N/A	DESTREG ← LSR SRCREG1
0x0E	BRA	IM	PC ← Value
0x0F	BEQ	IM	IF Z=1 THEN PC ← Value
0x10	BNE	IM	IF Z=0 THEN PC ← Value
0x11	CALL	IM	M[SP] ← PC, SP ← SP – 1, PC ← Value
0x12	RET	N/A	SP ← SP + 1, PC ← M[SP]

Figure 2: Opcode

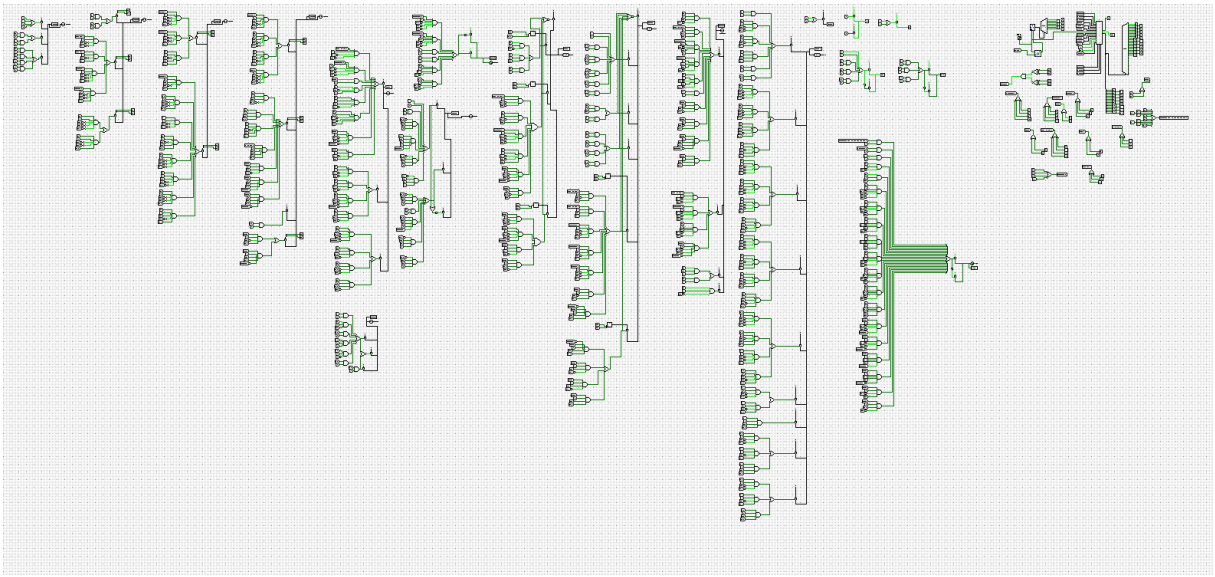

Figure 3: Full Circuit

2 PROJECT LOGIC

RF refers to General Purpose Register File
ARF refers to Address Register File

2.1 FETCH

T0: IR(8-15) ← M[PC]
OutDSel: 00; Select PC
Load: 1; Load M[PC]
L/H: 1; Select IR(8-15)
FunSelIR: 11; Load to chosen part of IR
En: 1;

T1: PC ← PC + 1
RegSelARF: 001; Only enable PC
FunSelARF: 10; Increment registers
OutDSel: 00; Select PC
En: 0; Disable IR

T2: IR(0-7) ← M[PC]
OutDSel: 00; Select PC
Load: 1; Load M[PC]
L/H: 0; Select IR(0-7)
FunSelIR: 11; Load to chosen part of IR
En: 1;

T3: PC ← PC + 1
RegSelARF: 001; Only enable PC
FunSelARF: 10; Increment registers
OutDSel: 00; Select PC
En: 0; Disable IR

2.2 DECODE

2.2.1 LD

D0: Rx ← Value IR(8) = 0, Addressing Mode is Immediate,

D0T4:
MuxASel: 00; Send IR(0-7) to RF
RegSelRF: decode(IR10,IR9); Enable given registers
FunSelRF: 01; Load to given registers

IR(8) = 1, Addressing Mode is Direct,
D0T4:
MuxBSel: 01; Send IR(0-7) to ARF
RegSelARF: 010; enable AR
FunSelARF: 01; Load to given registers

D0T5:
OutDSel: 01; Select AR
Load: 1; Load M[AR]
MuxASel: 01; Send M[AR] to RF
RegSelRF: decode(IR10,IR9); Enable given registers
FunSelRF: 01; Load to enabled registers

2.2.2 ST

D1: Value ← Rx
Addressing Mode is Direct
So, D1: M[AR] ← Rx

D1T4:
MuxBSel: 01; Send IR(0-7) to ARF
RegSelARF: 010; Enable AR
FunSelARF: 01; Load to chosen registers

D1T5:
OutDSel: 01; Select AR
OutASel: IR10, IR9; Select Rx
MuxCSel: 1 Select A input of ALU as Rx
FunSelALU: 0000; Select A (Rx) as output of ALU
Store: 1; Store Rx to M[AR]

2.2.3 MOV

D2: DESTREG ← SRCREG1

D2T4: IR10'IR7': RF ← RF
OutASel: (IR6,IR5); Select SRCREG1
FunSelALU: 0000; Select A (SRCREG1) as output of ALU
MuxASel: 11; Choose the SRCREG1
RegSelRF: decode(IR9,IR8); Enable the desired DESTREG
FunSelRF: 01; Load to DESTREG

D2T4: IR10'IR7: RF ← ARF
OutCSel: (IR6,IR5); Select SRCREG1
MuxASel: 10; Choose OutCSel as RF input
RegSelRF: decode(IR9,IR8); Enable the desired DESTREG
FunSelRF: 01; Load to DESTREG

D2T4: IR10IR7': ARF ← RF
OutASel: (IR6,IR5); Select SRCREG1
MuxCSel: 1; Choose OutASel output as A input of ALU
FunSelALU: 0000; Select A (SRCREG1) as output of ALU
MuxBSel: 11; Select RF as input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01 Load to DESTREG

D2T4: IR10IR7: ARF ← ARF
OutCSel: (IR6,IR5); Select SRCREG1
MuxASel: 10; Select SRCREG1 as I0 of MuxC
MuxCSel: 0; Select SRCREG1 as input A of ALU
FunSelALU: 0000; Select SRCREG1 as output A of ALU
MuxBSel: 11; Select SRCREG1 as input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load to DESTREG

2.2.4 PSH

D3: M[SP] ← Rx, SP ← SP - 1

D3T4: M[SP] ← RF
OutASel: (IR10,IR9); Select REGSEL
MuxCSel: 1; Send REGSEL as input A of ALU
FunSelALU: 0000; Send REGSEL as output of ALU

OutDSel: 10; Send SP as Address to M
Store: 1; Store REGSEL to M[SP]

D3T5:
RegSelARF: 100; Enable SP
FunSelARF: 11; Decrement SP

2.2.5 PUL

D4: SP ← SP + 1, Rx ← M[SP]

D4T4:
RegSelARF: 100; Enable SP
FunSelARF: 10; Incremenet SP

D4T5:
OutDSel: 10; Select SP as Address to M
Load: 1; Write M
MuxASel: 01; Send M[SP] to input of RF
RegSelRF: decode(IR10,IR9); Enable REGSEL
FunSelRF: 01; Load M[SP] to REGSEL

2.2.6 ADD

D5: DESTREG ← SRCREG1 + SRCREG2

D5T4: IR10'IR7'IR4': RF ← RF + RF
OutASel: (IR6,IR5); Select SRCREG1
MuxCSel: 1; Send SRCREG1 as input A of ALU
OutBSel: (IR3,IR2); Send SRCREG2 as input B of ALU
FunSelALU: 0100; Do the operation A + B
MuxASel: 11; Send ALU output to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D5T4: IR10'IR7'IR4: RF ← RF + ARF
LEFT FOR LATER (NEED PSH/PUL)

D5T4: IR10'IR7IR4': RF ← ARF + RF

LEFT FOR LATER (NEED PSH/PUL)

D5T4: IR10'IR7IR4: RF ← ARF + ARF
REDUNDANT

D5T4: IR10IR7'IR4': ARF ← RF + RF
OutASel: (IR6,IR5); Send SRCREG1 to MuxC
MuxCSel: 1; Send SRCREG1 to ALU
OutBSel: (IR3,IR2) Select SRCREG2
FunSelALU: 0100; Do the operation A + B
MuxBSel: 11; Send ALU output to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D5T4: IR10IR7'IR4: ARF ← RF + ARF
OutCSel: (IR3,IR2); Send SRCREG2 to MuxA
MuxASel: 10; Send SRCREG2 to MuxC
MuxCSel: 0; Send SRCREG2 as input A of ALU
OutBSel: (IR6,IR5); send SRCREG1 as input B of ALU
FunSelALU: 0100; Do the operation A + B
MuxBSel: 11; Send ALU output to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D5T4: IR10IR7IR4': ARF ← ARF + RF
OutCSel: (IR6,IR5); Send SRCREG1 to MuxA
MuxASel: 10; Send SRCREG1 to MuxC
MuxCSel: 0; Send SRCREG1 to A input of ALU
OutBSel: (IR3,IR2); Send SRCREG2 to B input of ALU
FunSelALU: 0100; Do the operation A + B
MuxBSel: 11; Send ALU output to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D5T4: IR10IR7IR4: ARF ← ARF + ARF
REDUNDANT

2.2.7 SUB

D6: DESTREG ←SRCREG2 - SRCREG1

D6T4: IR10'IR4'IR7': RF ← RF - RF
OutASel: (IR3,IR2); Send SRCREG2 to MuxC
MuxCSel: 1; Send SRCREG2 as input A of ALU
OutBSel: (IR6,IR5); Send SRCREG1 as input B of ALU
FunSelALU: 0110; Do the operation A - B
MuxASel: 11; Send output of ALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D6T4: IR10'IR4'IR7: RF ← RF - ARF
REDUNDANT

D6T4: IR10'IR4IR7': RF ← ARF - RF
OutBSel: (IR6,IR5); Select SRCREG1 as B input of ALU
OutCSel: (IR3,IR2); Send SRCREG2 to MuxA
MuxASel: 10; Send SRCREG2 to MuxC
MuxCSel: 0; Select SRCREG1 as A input of ALU

D6T5:
FunSelALU: 0110; Do the operation A - B
MuxASel: 11; Send ALU output to RF input
RegSelRF: decode(IR9,IR8); enable DESTREG
FunSelRF: 01; Load result to DESTREG

D6T4: IR10'IR4IR7: RF ← ARF - ARF
REDUNDANT

D6T4: IR10IR4'IR7': ARF ← RF - RF
OutASel: (IR3,IR2); Send SRCREG2 to MuxC
MuxCSel: 1; Select SRCREG2 as A input of ALU
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU; 0110; Do the operation A - B
MuxBSel; 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG

FunSelARF: 01; Load result to DESTREG

D6T4: IR10IR4'IR7: ARF ← RF - ARF
REDUNDANT

D6T4: IR10IR4IR7': ARF ← ARF - RF
OutCSel: (IR3,IR2); Select SRCREG2 to MuxA
MuxASel: 10; Send SRCREG 2 to MuxC
MuxCSel: 0; Send SRCREG as A input of ALU
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU: 0110; Do the operation A - B
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D6T4: IR10IR4IR7: ARF ← ARF - ARF
REDUNDANT

2.2.8 DEC

DESTREG ← SRCREG1 - 1

D7T4: IR10'IR7': IR9IR8 = IR6IR5: RF ← RF - 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 11; Decrement SRCREG1 by 1

D7T5: Fixing flag
OutBSel: (IR6,IR5); Send SRCREG1 to B input of ALU
FunSelALU: 0001; Send B as outALU

D7T4: IR10'IR7': IR9IR8 != IR6IR5: RF ← RF - 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 11; Decrement SRCREG1 by 1

D7T5:
OutBSel: (IR6,IR5); Send SRCREG1 (decremented) to B input of ALU
FunSelALU: 0001; Print outALU as SRCREG1 (decremented)
MuxASel: 11; Send outALU to RF input

RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG
D7T6:
OutBSel: (IR9,IR8); Send DESTREG to ALU B input
FunSelALU: 0001; Print outALU as SRCREG1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 10; Increment SRCREG1 back to its original value

D7T4: IR10'IR7: RF ← ARF - 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 11; Decrement SRCREG1 by 1

D7T5:
OutCSel: (IR6,IR5); Send SRCREG1 (decremented) to MuxA
MuxASel: 10; Send SRCREG1 to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load to DESTREG

D7T6:
OutBSel: (IR9,IR8); Send DESTREG to B input of ALU
FunSelALU: 0001; Print outALU as DESTREG
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 10; Increment SRCREG1 back to its original value

D7T4: IR10IR7': ARF ← RF - 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 11; Decrement SRCREG1 by 1

D7T5:
OutBSel: (IR6,IR5); Send SRCREG1 to B input of ALU
FunSelALU: 0001; Print SRCREG1 to outALU
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D7T6:
RegSelRF: decode(IR6,IR6); Enable SRCREG1
FunSelRF: 10; Increment SRCREG1 back to its original value

OutCSel: (IR9,IR8); Send DESTREG to MuxA
MuxASel: 10; Send DESTREG to MuxC
MuxCSel: 0; Send DESTREG to A input of ALU
FunSelALU: 0000; print DESTREG to outALU

D7T4: IR10IR7: IR9IR8 = IR6IR5: ARF ← ARF - 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 11; Decrement SRCREG1 by 1

D7T5:
OutCSel: (IR6,IR5); Send SRCREG1/DESTREG to MuxA
MuxASel: 10; Send SRCREG1/DESTREG to MuxCSel
MuxCSel: 0; Send SRCREG1/DESTREG to input of A of ALU
FunSelALU: 0000; Print SRCREG1/DESTREG to outALU

D7T4: IR10IR7: IR9IR8 != IR6IR5: ARF ← ARF - 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 11; Decrement SRCREG1 by 1

D7T5:
OutCSel: (IR6,IR5); Send SRCREG1 (decremented) to MuxA
MuxASel: 10; Send SRCREG1 to MuxC
MuxCSel: 0; Send SRCREG1 to A input of ALU
FunSelALU: 0000; Print SRCREG1 as outALU
MuxBSel: 11; Send SRCREG1 to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load to DESTREG

D7T6:
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 10; Increment SRCREG1 back to its original value
OutCSel: (IR9,IR8); Send DESTREG to MuxA
MuxASel: 10; Send DESTREG to MuxCSel
MuxCSel: 0; Send DESTREG to A input of ALU
FunSelALU: 0000; Print DESTREG as outALU

2.2.9 INC

DESTREG ← SRCREG1 + 1

D8T4: IR10'IR7': IR9IR8 = IR6IR5: RF ← RF + 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 10; Increment SRCREG1 by 1

D8T5: Fixing flag
OutBSel: (IR6,IR5); Send SRCREG1 to B input of ALU
FunSelALU: 0001; Send B as outALU

D8T4: IR10'IR7': IR9IR8 != IR6IR5: RF ← RF + 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 10; Increment SRCREG1 by 1

D8T5:
OutBSel: (IR6,IR5); Send SRCREG1 (incremented) to B input of ALU
FunSelALU: 0001; Print outALU as SRCREG1 (incremented)
MuxASel: 11;
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG
D8T6:
OutBSel: (IR9,IR8); Send DESTREG to ALU B input
FunSelALU: 0001; Print outALU as SRCREG1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 11; Decrement SRCREG1 back to its original value

D8T4: IR10'IR7: RF ← ARF + 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 10; Increment SRCREG1 by 1

D8T5:
OutCSel: (IR6,IR5); Send SRCREG1 (incremented) to MuxA
MuxASel: 10; Send SRCREG1 to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load to DESTREG

D8T6:
OutBSel: (IR9,IR8); Send DESTREG to B input of ALU
FunSelALU: 0001; Print outALU as DESTREG
RegSelARF: decode(IR6,IR5); Enable SRCREG1

FunSelARF: 11; Decrement SRCREG1 back to its original value

D8T4: IR10IR7': ARF ← RF + 1
RegSelRF: decode(IR6,IR5); Enable SRCREG1
FunSelRF: 10; Increment SRCREG1 by 1

D8T5:
OutBSel: (IR6,IR5); Send SRCREG1 to B input of ALU
FunSelALU: 0001; Print SRCREG1 to outALU
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D8T6:
RegSelRF: decode(IR6,IR6); Enable SRCREG1
FunSelRF: 11; Decrement SRCREG1 back to its original value
OutCSel: (IR9,IR8); Send DESTREG to MuxA
MuxASel: 10; Send DESTREG to MuxC
MuxCSel: 0; Send DESTREG to A input of ALU
FunSelALU: 0000; print DESTREG to outALU

D8T4: IR10IR7: IR9IR8 = IR6IR5: ARF ← ARF + 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 10; Increment SRCREG1 by 1

D8T5:
OutCSel: (IR6,IR5); Send SRCREG1/DESTREG to MuxA
MuxASel: 10; Send SRCREG1/DESTREG to MuxCSel
MuxCSel: 0; Send SRCREG1/DESTREG to input of A of ALU
FunSelALU: 0000; Print SRCREG1/DESTREG to outALU

D8T4: IR10IR7: IR9IR8 != IR6IR5: ARF ← ARF + 1
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 10; Increment SRCREG1 by 1

D8T5:
OutCSel: (IR6,IR5); Send SRCREG1 (incremented) to MuxA
MuxASel: 10; Send SRCREG1 to MuxC

MuxCSel: 0; Send SRCREG1 to A input of ALU
FunSelALU: 0000; Print SRCREG1 as outALU
MuxBSel: 11; Send SRCREG1 to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load to DESTREG

D8T6:
RegSelARF: decode(IR6,IR5); Enable SRCREG1
FunSelARF: 11; Decrement SRCREG1 back to its original value
OutCSel: (IR9,IR8); Send DESTREG to MuxA
MuxASel: 10; Send DESTREG to MuxCSel
MuxCSel: 0; Send DESTREG to A input of ALU
FunSelALU: 0000; Print DESTREG as outALU

2.2.10 AND

DESTREG ← SRCREG1 AND SRCREG2

D9T4: IR10'IR7'IR4': RF ← RF AND RF
OutASel: (IR6,IR5); Send SRCREG1 to MuxC
MuxCSel: 1; Send SRCREG1 as A input of ALU
OutBSel: (IR3,IR2); Send SRCREG2 as B input of ALU
FunSelALU: 0111; Do the operation A AND B
MuxASel: 11; Send outALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D9T4: IR10'IR7'IR4: RF ← RF AND ARF
LEFT FOR LATER (NEED PSH/PUL)

D9T4: IR10'IR7IR4': RF ← ARF AND RF
LEFT FOR LATER (NEED PSH/PUL)

D9T4: IR10'IR7IR4: RF ← ARF AND ARF
REDUNDANT

D9T4: IR10IR7'IR4': ARF ← RF AND RF
OutASel: (IR6,IR5); Send SRCREG1 to MuxC

MuxCSel: 1; Send SRCREG1 to A input of ALU
OutBSel: (IR3,IR2); Send SRCREG2 to B input of ALU
FunSelALU: 0111; Do the operation A AND B
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D9T4: IR10IR7'IR4: ARF ← RF AND ARF
OutCSel: (IR3,IR2); Send SRCREG2 to MuxA
MuxASel: 10; Send SRCREG2 to MuxC
MuxCSel: 0; Send SRCREG2 as A input of ALU
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU; 0111; Do the operation A AND B
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D9T4: IR10IR7IR4': ARF ← ARF AND RF
SAME AS ABOVE (IR10IR7'IR4: ARF ← RF AND ARF)

D9T4: IR10IR7IR4: ARF ← ARF AND ARF
REDUNDANT

2.2.11 OR

DESTREG ← SRCREG1 OR SRCREG2

D10T4: IR10'IR7'IR4': RF ← RF OR RF
OutASel: (IR6,IR5); Send SRCREG1 to MuxC
MuxCSel: 1; Send SRCREG1 as A input of ALU
OutBSel: (IR3,IR2); Send SRCREG2 as B input of ALU
FunSelALU: 1000; Do the operation A AND B
MuxASel: 11; Send outALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D10T4: IR10'IR7'IR4: RF ← RF OR ARF
LEFT FOR LATER (NEED PSH/PUL)

D10T4: IR10'IR7IR4': RF ← ARF OR RF
LEFT FOR LATER (NEED PSH/PUL)

D10T4: IR10'IR7IR4: RF ← ARF OR ARF
REDUNDANT

D10T4: IR10IR7'IR4': ARF ← RF OR RF
OutASel: (IR6,IR5); Send SRCREG1 to MuxC
MuxCSel: 1; Send SRCREG1 to A input of ALU
OutBSel: (IR3,IR2); Send SRCREG2 to B input of ALU
FunSelALU: 1000; Do the operation A AND B
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D10T4: IR10IR7'IR4: ARF ← RF OR ARF
OutCSel: (IR3,IR2); Send SRCREG2 to MuxA
MuxASel: 10; Send SRCREG2 to MuxC
MuxCSel: 0; Send SRCREG2 as A input of ALU
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU; 1000; Do the operation A AND B
MuxBSel: 11; Send outALU to ARF input
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D10T4: IR10IR7IR4': ARF ← ARF OR RF
SAME AS ABOVE (IR10IR7'IR4: ARF ← RF AND ARF)

D10T4: IR10IR7IR4: ARF ← ARF OR ARF
REDUNDANT

2.2.12 NOT

DESTREG ← NOT SRCREG1

D11T4: IR10'IR7' : RF ← RF
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU; 0011; Do the operation -B

MuxASel: 11; Send outALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D11T4: IR10'IR7 : RF ← ARF
LEFT FOR LATER (NEED PSH/PUL)

D11T4: IR10IR7 : ARF ← ARF
OutCSel: (IR6,IR5); Select SRCREG1 to MuxA
MuxASel: 10; Send SRCREG1 to MuxC
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU: 0010; Do the operation -A
MuxBSel: 11; Send outALU to input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D11T4: IR10IR7' : ARF ← RF
OutBSel: (IR6,IR5); Send SRCREG1 as B input of ALU
FunSelALU; 0011; Do the operation -B
MuxBSel: 11; Send outALU to input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

2.2.13 LSL

DESTREG ← LSL SRCREG1

D12T4: IR10'IR7' : RF ← RF
OutASel: (IR6,IR5); Send SRCREG1 as A input of ALU
MuxCSel: 1; Send SRCREG1 as A input of ALU
FunSelALU; 1010 ; Do the operation LSL
MuxASel: 11; Send outALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D12T4: IR10'IR7 : RF ← ARF
LEFT FOR LATER (NEED PSH/PUL)

D12T4: IR10IR7 : ARF ← ARF
OutCSel: (IR6,IR5); Select SRCREG1 to MuxA
MuxASel: 10; Send SRCREG1 to MuxC
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU; 1010 ; Do the operation LSL
MuxBSel: 11; Send outALU to input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D12T4: IR10IR7' : ARF ← RF
OutASel: (IR6,IR5); Send SRCREG1 as A input of ALU
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU; 1010 ; Do the operation LSL
MuxBSel: 11; Send outALU to input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

2.2.14 LSR

DESTREG ← LSR SRCREG1

D13T4: IR10'IR7' : RF ← RF
OutASel: (IR6,IR5); Send SRCREG1 as A input of ALU
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU; 1011 ; Do the operation LSL
MuxASel: 11; Send outALU to RF input
RegSelRF: decode(IR9,IR8); Enable DESTREG
FunSelRF: 01; Load result to DESTREG

D13T4: IR10'IR7 : RF ← ARF
LEFT FOR LATER (NEED PSH/PUL)

D13T4: IR10IR7 : ARF ← ARF
OutCSel: (IR6,IR5); Select SRCREG1 to MuxA
MuxASel: 10; Send SRCREG1 to MuxC
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU; 1011 ; Do the operation LSL
MuxBSel: 11; Send outALU to input of ARF

RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

D13T4: IR10IR7' : ARF ← RF
OutASel: (IR6,IR5); Send SRCREG1 as A input of ALU
MuxCSel: 0; Send SRCREG1 as A input of ALU
FunSelALU; 1011 ; Do the operation LSL
MuxBSel: 11; Send outALU to input of ARF
RegSelARF: decode(IR9,IR8); Enable DESTREG
FunSelARF: 01; Load result to DESTREG

2.2.15 BRA

PC ← Value
Addressing Mode is Immediate, So,
PC ← IR(7-0)

D14T4:
MuxBSel: 01; Send IR(7-0) to ARF input
RegSelARF: 001; Enable PC
FunSelARF: 01; Load IR(7-0) to PC

2.2.16 BEQ

IF Z=1 THEN PC ← Value
Addressing Mode is Immediate, So,
IF Z=1 THEN PC ← IR(7-0)

D15T4: Z': DO NOTHING

D15T4: Z:
MuxBSel: 01; Send IR(7-0) to ARF input
RegSelARF: 001; Enable PC
FunSelARF: 01; Load IR(7-0) to PC

2.2.17 BNE

IF Z=0 THEN PC ← Value
Addressing Mode is Immediate, So,
IF Z=0 THEN PC ← IR(7-0)

D16T4: Z': DO NOTHING

D16T4: Z':
MuxBSel: 01; Send IR(7-0) to ARF input
RegSelARF: 001; Enable PC
FunSelARF: 01; Load IR(7-0) to PC

2.2.18 CALL

M[SP] ← PC, SP ← SP – 1, PC ← Value
Addressing Mode is Immediate, So,
M[SP] ← PC, SP ← SP – 1, PC ← IR(7-0)

D17T4:
OutCSel: 00; Send PC to MuxA
MuxASel: 10; Send PC to MuxC
MuxCSel: 0; Send PC as input A of ALU
FunSelALU: 0000; Select PC as output of ALU
OutDSel: 10; Send SP to address of M
Store: 1;

D17T5:
RegSelARF: 100; Enable SP
FunSelARF: 11; Decrement SP by 1

D17T6:
MuxBSel: 01; Send IR(7-0) to ARF input
RegSelARF: 001; Enable PC
FunSelARF: 01; Load IR(7-0) to PC

2.2.19 RET

SP ← SP + 1, PC ← M[SP]

D18T4:
RegSelARF: 100; Enable SP
FunSelARF: 10; Increment SP by 1

D18T5:
OutDSel: 10; Select SP as the address of M

Load: 1; Load M[SP]
MuxBSel: 10; Send M[SP] to ARF input
RegSelARF: 001; Enable PC
FunSelARF: 01; Load M[SP] to PC

3 PROJECT EQUATIONS

3.1 ORIGINAL

We firstly implemented our circuit without the INC and DEC operations

3.1.1 OutASel

D1T5: IR10, IR9;
D2T4: IR10'IR7': (IR6,IR5);
D2T4: IR10IR7': (IR6,IR5)
D3T4: (IR10,IR9)
D5T4: IR10'IR7'IR4': (IR6,IR5)
D5T4: IR10IR7'IR4': (IR6,IR5)
D6T4: IR10'IR4'IR7': (IR3,IR2)
D6T4: IR10IR4'IR7': (IR3,IR2);
D9T4: IR10'IR7'IR4': (IR6,IR5);
D9T4: IR10IR7'IR4': (IR6,IR5);
D10T4: IR10'IR7'IR4': (IR6,IR5);
D10T4: IR10IR7'IR4': (IR6,IR5)
D12T4: IR10'IR7': (IR6,IR5)
D12T4: IR10IR7' :(IR6,IR5)
D13T4: IR10'IR7' : (IR6,IR5);
D13T4: IR10IR7': (IR6,IR5);

3.1.2 OutBSel

D5T4: IR10'IR7'IR4': (IR3,IR2);
D5T4: IR10IR7'IR4': (IR3,IR2)
D5T4: IR10IR7'IR4: (IR6,IR5)
D5T4: IR10IR7IR4': (IR3,IR2)
D6T4: IR10'IR4'IR7': (IR6,IR5)
D6T4: IR10'IR4IR7': (IR6,IR5)
D6T4: IR10IR4'IR7': (IR6,IR5);

D6T4: IR10IR4IR7': (IR6,IR5);
D9T4: IR10'IR7'IR4': (IR3,IR2);
D9T4: IR10IR7'IR4': (IR3,IR2);
D9T4: IR10IR7'IR4: (IR6,IR5);
D9T4: IR10IR7IR4': (IR6,IR5);
D10T4: IR10'IR7'IR4': (IR3,IR2);
D10T4: IR10IR7'IR4': (IR3,IR2);
D10T4: IR10IR7'IR4: (IR6,IR5);
D10T4: IR10IR7IR4': (IR6,IR5);
D11T4: IR10'IR7' : (IR6,IR5);
D11T4: IR10IR7': (IR6,IR5);

3.1.3 RegSelRF

D0T4: IR(8) = 0, decode(IR10,IR9);
D0T5: IR(8) = 1, decode(IR10,IR9)
D2T4: IR10'IR7': decode(IR9,IR8)
D2T4: IR10'IR7: decode(IR9,IR8)
D4T5: decode(IR10,IR9)
D5T4: IR10'IR7'IR4': decode(IR9,IR8)
D6T4: IR10'IR4'IR7': decode(IR9,IR8);
D6T5: IR10'IR4IR7': decode(IR9,IR8);
D9T4: IR10'IR7'IR4': decode(IR9,IR8)
D10T4: IR10'IR7'IR4': decode(IR9,IR8);
D11T4: IR10'IR7' : decode(IR9,IR8)
D12T4: IR10'IR7' : decode(IR9,IR8);
D13T4: IR10'IR7' : decode(IR9,IR8);

3.1.4 FunSelRF

D0T4: IR(8) = 0, 01;
D0T5: 01;
D2T4: IR10'IR7': 01;
D2T4: IR10'IR7: 01;
D4T5: 01;
D5T4: IR10'IR7'IR4': 01;
D6T4: IR10'IR4'IR7': 01;
D6T5: IR10'IR4IR7': 01
D9T4: IR10'IR7'IR4': 01;

D10T4: IR10'IR7'IR4': 01;
D11T4: IR10'IR7' : 01;
D12T4: IR10'IR7' : 01;
D13T4: IR10'IR7' : 01;

3.1.5 MuxCSel

D1T5: 1
D2T4: IR10IR7': 1
D2T4: IR10IR7: 0;
D3T4: 1;
D5T4: IR10'IR7'IR4': 1;
D5T4: IR10IR7'IR4': 1;
D5T4: IR10IR7'IR4: 0;
D5T4: IR10IR7IR4': 0;
D6T4: IR10'IR4'IR7': 1;
D6T4: IR10'IR4IR7': 0;
D6T4: IR10IR4'IR7': 1;
D6T4: IR10IR4IR7': 0;
D9T4: IR10'IR7'IR4': 1;
D9T4: IR10IR7'IR4': 1;
D9T4: IR10IR7'IR4: 0;
D9T4: IR10IR7IR4': 0;
D10T4: IR10'IR7'IR4': 1;
D10T4: IR10IR7'IR4': 1;
D10T4: IR10IR7'IR4: 0;
D10T4: IR10IR7IR4': 0;
D11T4: IR10IR7 : 0;
D12T4: IR10'IR7' : 1;
D12T4: IR10IR7 : 0;
D12T4: IR10IR7' : 0;
D13T4: IR10'IR7' : 0;
D13T4: IR10IR7 : 0;
D13T4: IR10IR7' : 0;
D17T4: 0;

3.1.6 FunSelALU

D1T5: 0000;

D2T4: IR10'IR7': 0000;
D2T4: IR10IR7': 0000;
D2T4: IR10IR7: 0000;
D3T4: 0000;
D5T4: IR10'IR7'IR4': 0100;
D5T4: IR10IR7'IR4': 0100;
D5T4: IR10IR7'IR4: 0100;
D5T4: IR10IR7IR4': 0100;
D6T4: IR10'IR4'IR7': 0110;
D6T5: IR10'IR4IR7': 0110;
D6T4: IR10IR4'IR7': 0110;
D6T4: IR10IR4IR7': 0110;
D9T4: IR10'IR7'IR4': 0111;
D9T4: IR10IR7'IR4': 0111;
D9T4: IR10IR7'IR4: 0111;
D9T4: IR10IR7IR4': 0111;
D10T4: IR10'IR7'IR4': 1000;
D10T4: IR10IR7'IR4': 1000;
D10T4: IR10IR7'IR4: 1000;
D10T4: IR10IR7IR4': 1000;
D11T4: IR10'IR7' : 0011;
D11T4: IR10IR7 : 0010;
D11T4: IR10IR7' : 0011;
D12T4: IR10'IR7' : 1010;
D12T4: IR10IR7 : 1010;
D12T4: IR10IR7' : 1010;
D13T4: IR10'IR7' : 1011;
D13T4: IR10IR7 : 1011;
D13T4: IR10IR7' : 1011;
D17T4: 0000;

3.1.7 MuxASel

IR(8) = 0, D0T4: 00;
IR(8) = 1, D0T5: 01;
D2T4: IR10'IR7': 11;
D2T4: IR10'IR7: 10;
D2T4: IR10IR7: 10;

D4T5: 01;
D5T4: IR10'IR7'IR4': 11;
D5T4: IR10IR7'IR4: 10;
D5T4: IR10IR7IR4': 10;
D6T4: IR10'IR4'IR7': 11;
D6T4: IR10'IR4IR7': 10;
D6T5: IR10'IR4IR7': 11;
D6T4: IR10IR4IR7': 10;
D9T4: IR10'IR7'IR4': 11;
D9T4: IR10IR7'IR4: 10;
D9T4: IR10IR7IR4': 10;
D10T4: IR10'IR7'IR4': 11;
D10T4: IR10IR7'IR4: 10;
D10T4: IR10IR7IR4': 10;
D11T4: IR10'IR7' : 11;
D11T4: IR10IR7 : 10;
D12T4: IR10'IR7' : 11;
D12T4: IR10IR7 : 10;
D13T4: IR10'IR7' : 11;
D13T4: IR10IR7 : 10;
D17T4: 10;

3.1.8 En

T0: 1;
T1: 0;
T2: 1;
T3: 0;

3.1.9 L/H

T0: 1;
T2: 0;

3.1.10 FunSelIR

T0: 11;
T2: 11;

3.1.11 MuxBSel

D0T4: IR(8) = 1, 01;
D1T4: 01;
D2T4: IR10IR7': 11;
D2T4: IR10IR7: 11;
D5T4: IR10IR7'IR4': 11;
D5T4: IR10IR7'IR4: 11;
D5T4: IR10IR7IR4': 11;
D6T4: IR10IR4'IR7': 11;
D6T4: IR10IR4IR7': 11;
D9T4: IR10IR7'IR4': 11;
D9T4: IR10IR7'IR4: 11;
D9T4: IR10IR7IR4': 11;
D10T4: IR10IR7'IR4': 11;
D10T4: IR10IR7'IR4: 11;
D10T4: IR10IR7IR4': 11;
D11T4: IR10IR7 : 11;
D11T4: IR10IR7' : 11;
D12T4: IR10IR7 : 11;
D12T4: IR10IR7' : 11;
D13T4: IR10IR7 : 11;
D13T4: IR10IR7' : 11;
D14T4: 01;
D15T4: Z: 01
D16T4: Z': 01;
D17T6: 01;
D18T5: 10;

3.1.12 Store

D1T5: 1;
D3T4: 1;
D17T4: 1;

3.1.13 Load

T0: 1;
T2: 1;

D0T5: 1;
D4T5: 1;
D18T5: 1;

3.1.14 FunSelARF

T1: 10;
T3: 10;
D0T4: IR(8) = 1, 01;
D1T4: 01;
D2T4: IR10IR7': 01;
D2T4: IR10IR7: 01;
D3T5: 11;
D4T4: 10;
D5T4: IR10IR7'IR4': 01;
D5T4: IR10IR7'IR4: 01;
D5T4: IR10IR7IR4': 01;
D6T4: IR10IR4'IR7': 01;
D6T4: IR10IR4IR7': 01;
D9T4: IR10IR7'IR4': 01;
D9T4: IR10IR7'IR4: 01;
D9T4: IR10IR7IR4': 01;
D10T4: IR10IR7'IR4': 01;
D10T4: IR10IR7'IR4: 01;
D10T4: IR10IR7IR4': 01;
D11T4: IR10IR7 : 01;
D11T4: IR10IR7' : 01;
D12T4: IR10IR7 : 01;
D12T4: IR10IR7' : 01;
D13T4: IR10IR7 : 01;
D13T4: IR10IR7' : 01;
D14T4: 01;
D15T4: Z: 01;
D16T4: Z': 01;
D17T5: 11; 01;
D17T6: 01; 01;
D18T4: 10; 01;
D18T5: 01; 01;

3.1.15 RegSelARF

T1: 001;
T3: 001;
D0T4: IR(8) = 1, 010;
D1T4: 010;
D2T4: IR10IR7': decode(IR9,IR8);
D2T4: IR10IR7: decode(IR9,IR8);
D3T5: 100;
D4T4: 100;
D5T4: IR10IR7'IR4': decode(IR9,IR8);
D5T4: IR10IR7'IR4: decode(IR9,IR8);
D5T4: IR10IR7IR4': decode(IR9,IR8);
D6T4: IR10IR4'IR7': decode(IR9,IR8);
D6T4: IR10IR4IR7': decode(IR9,IR8);
D9T4: IR10IR7'IR4': decode(IR9,IR8);
D9T4: IR10IR7'IR4: decode(IR9,IR8);
D9T4: IR10IR7IR4': decode(IR9,IR8);
D10T4: IR10IR7'IR4': decode(IR9,IR8);
D10T4: IR10IR7'IR4: decode(IR9,IR8);
D10T4: IR10IR7IR4': decode(IR9,IR8);
D11T4: IR10IR7 : decode(IR9,IR8);
D11T4: IR10IR7' : decode(IR9,IR8);
D12T4: IR10IR7 : decode(IR9,IR8);
D12T4: IR10IR7' : decode(IR9,IR8);
D13T4: IR10IR7 : decode(IR9,IR8);
D13T4: IR10IR7' : decode(IR9,IR8);
D14T4: 001;
D15T4: Z: 001;
D16T4: Z': 001;
D17T5: 100;
D17T6: 001;
D18T4: 100;
D18T5: 001;

3.1.16 OutCSel

D2T4: IR10'IR7: (IR6,IR5);

D2T4: IR10IR7: (IR6,IR5);
D5T4: IR10IR7'IR4: (IR3,IR2);
D5T4: IR10IR7IR4': (IR6,IR5);
D6T4: IR10'IR4IR7': (IR3,IR2);
D6T4: IR10IR4IR7': (IR3,IR2);
D9T4: IR10IR7'IR4: (IR3,IR2);
D9T4: IR10IR7IR4': (IR3,IR2);
D10T4: IR10IR7'IR4: (IR3,IR2);
D10T4: IR10IR7IR4': (IR3,IR2);
D11T4: IR10IR7 : (IR6,IR5);
D12T4: IR10IR7 : (IR6,IR5);
D13T4: IR10IR7 : (IR6,IR5);
D17T4: 00;

3.1.17 OutDSel

T0: 00;
T1: 00;
T2: 00;
T3: 00;
D0T5: 01;
D1T5: 01;
D3T4: 10;
D4T5: 10;
D17T4; 10;
D18T5: 10;

3.1.18 SC Clear

D0T4IR8'
D0T5IR8
D1T5
D2T4
D3T5
D4T5
D5T4
D6T4IR10'IR4'IR7'
D6T4IR10'IR4'IR7
D6T5IR10'IR4IR7'

D6T4IR10'IR4IR7
D6T4IR10IR4'IR7'
D6T4IR10IR4'IR7
D6T4IR10IR4IR7'
D6T4IR10IR4IR7
D7T5: IR10'IR7': SRC==DEST
D7T5: IR10'IR7': !(SRC==DEST)
D7T6: IR10'IR7
D7T6: IR10IR7'
D7T5: IR10IR7: SRC==DEST
D7T6: IR10IR7: !(SRC==DEST)
D8T5: IR10'IR7': SRC==DEST
D8T5: IR10'IR7': !(SRC==DEST)
D8T6: IR10'IR7
D8T6: IR10IR7'
D8T5: IR10IR7: SRC==DEST
D8T6: IR10IR7: !(SRC==DEST)
D9T4
D10T4
D11T4
D12T4
D13T4
D14T4
D15T4
D16T4
D17T6
D18T5

3.2 COMPLETE

We managed to find a way to implement INC and DEC operations.

3.2.1 RegSelRF

D7T4: IR10'IR7': decode(IR6,IR5);
D7T5: IR10'IR7': IR9IR8 != IR6IR5: decode(IR9,IR8);
D7T6: IR10'IR7': IR9IR8 != IR6IR5: decode(IR6,IR5);
D7T5: IR10'IR7: decode(IR9,IR8);

D7T4: IR10IR7': decode(IR6,IR5);
D7T6: IR10IR7': decode(IR6,IR6);
D8T4: IR10'IR7': decode(IR6,IR5);
D8T5: IR10'IR7': IR9IR8 != IR6IR5: decode(IR9,IR8);
D8T6: IR10'IR7': IR9IR8 != IR6IR5: decode(IR6,IR5);
D8T5: IR10'IR7: decode(IR9,IR8);
D8T4: IR10IR7': decode(IR6,IR5);
D8T6: IR10IR7': decode(IR6,IR6);

3.2.2 FunSelRF

D7T4: IR10'IR7': 11;
D7T5: IR10'IR7': IR9IR8 != IR6IR5: 01;
D7T6: IR10'IR7': IR9IR8 != IR6IR5: 10;
D7T5: IR10'IR7: 01;
D7T4: IR10IR7': 11;
D7T6: IR10IR7': 10;
D8T4: IR10'IR7': 10;
D8T5: IR10'IR7': IR9IR8 != IR6IR5: 01;
D8T6: IR10'IR7': IR9IR8 != IR6IR5: 11;
D8T5: IR10'IR7: 01;
D8T4: IR10IR7': 10;
D8T6: IR10IR7': 11;

3.2.3 OutBSel

D7T5: IR10'IR7': (IR6,IR5);
D7T6: IR10'IR7': IR9IR8 != IR6IR5: (IR9,IR8);
D7T6: IR10'IR7: (IR9,IR8);
D7T5: IR10IR7': (IR6,IR5);
D8T5: IR10'IR7': (IR6,IR5);
D8T6: IR10'IR7': IR9IR8 != IR6IR5: (IR9,IR8);
D8T6: IR10'IR7: (IR9,IR8);
D8T5: IR10IR7': (IR6,IR5);

3.2.4 FunSelALU

D7T5: IR10'IR7': 0001;
D7T6: IR10'IR7': IR9IR8 != IR6IR5: 0001;

D7T6: IR10'IR7: 0001;
D7T5: IR10IR7': 0001;
D7T6: IR10IR7': 0000;
D7T5: IR10IR7: 0000;
D7T6: IR10IR7: IR9IR8 != IR6IR5: 0000;
D8T5: IR10'IR7': 0001;
D8T6: IR10'IR7': IR9IR8 != IR6IR5: 0001;
D8T6: IR10'IR7: 0001;
D8T5: IR10IR7': 0001;
D8T6: IR10IR7': 0000;
D8T5: IR10IR7: 0000;
D8T6: IR10IR7: IR9IR8 != IR6IR5: 0000;

3.2.5 MuxASel

D7T5: IR10'IR7': IR9IR8 != IR6IR5: 11;
D7T5: IR10'IR7: 10;
D7T6: IR10IR7': 10;
D7T5: IR10IR7: IR9IR8 = IR6IR5:10;
D7T5: IR10IR7: IR9IR8 != IR6IR5: 10;
D7T6: IR10IR7: IR9IR8 != IR6IR5: 10;
D8T5: IR10'IR7': IR9IR8 != IR6IR5: 11;
D8T5: IR10'IR7: 10;
D8T6: IR10IR7': 10;
D8T5: IR10IR7: IR9IR8 = IR6IR5:10;
D8T5: IR10IR7: IR9IR8 != IR6IR5: 10;
D8T6: IR10IR7: IR9IR8 != IR6IR5: 10;

3.2.6 FunSelARF

D7T4: IR10'IR7: 11;
D7T6: IR10'IR7: 10;
D7T5: IR10IR7': 01;
D7T4: IR10IR7: 11;
D7T5: IR10IR7: IR9IR8 != IR6IR5: 01;
D7T6: IR10IR7: IR9IR8 != IR6IR5: 10;
D8T4: IR10'IR7: 10;
D8T6: IR10'IR7: 11;
D8T5: IR10IR7': 01;

D8T4: IR10IR7: 10;
D8T5: IR10IR7: IR9IR8 != IR6IR5: 01;
D8T6: IR10IR7: IR9IR8 != IR6IR5: 11;

3.2.7 OutCSel

D7T5: IR10'IR7: (IR6,IR5);
D7T6: IR10IR7': (IR9,IR8);
D7T4: IR10IR7: IR9IR8 = IR6IR5: (IR6,IR5);
D7T5: IR10IR7: IR9IR8 != IR6IR5: (IR6,IR5);
D7T6: IR10IR7: IR9IR8 != IR6IR5: (IR9,IR8);
D8T5: IR10'IR7: (IR6,IR5);
D8T6: IR10IR7': (IR9,IR8);
D8T4: IR10IR7: IR9IR8 = IR6IR5: (IR6,IR5);
D8T5: IR10IR7: IR9IR8 != IR6IR5: (IR6,IR5);
D8T6: IR10IR7: IR9IR8 != IR6IR5: (IR9,IR8);

3.2.8 RegSelARF

D7T4: IR10'IR7: decode(IR6,IR5);
D7T6: IR10'IR7: decode(IR6,IR5);
D7T5: IR10IR7': decode(IR9,IR8);
D7T4: IR10IR7: decode(IR6,IR5);
D7T5: IR10IR7: IR9IR8 != IR6IR5: decode(IR9,IR8);
D7T6: IR10IR7: IR9IR8 != IR6IR5: decode(IR6,IR5);
D8T4: IR10'IR7: decode(IR6,IR5);
D8T6: IR10'IR7: decode(IR6,IR5);
D8T5: IR10IR7': decode(IR9,IR8);
D8T4: IR10IR7: decode(IR6,IR5);
D8T5: IR10IR7: IR9IR8 != IR6IR5: decode(IR9,IR8);
D8T6: IR10IR7: IR9IR8 != IR6IR5: decode(IR6,IR5);

3.2.9 MuxBSel

D7T5: IR10IR7': 11;
D7T5: IR10IR7: IR9IR8 != IR6IR5: 11;
D8T5: IR10IR7': 11;
D8T5: IR10IR7: IR9IR8 != IR6IR5: 11;

3.2.10 MuxCSel

D7T6: IR10IR7': 0;

D7T5: IR10IR7: 0;

D7T6: IR10IR7: IR9IR8 != IR6IR5: 0;

D8T6: IR10IR7': 0;

D8T5: IR10IR7: 0;

D8T6: IR10IR7: IR9IR8 != IR6IR5: 0;

4 RESULTS

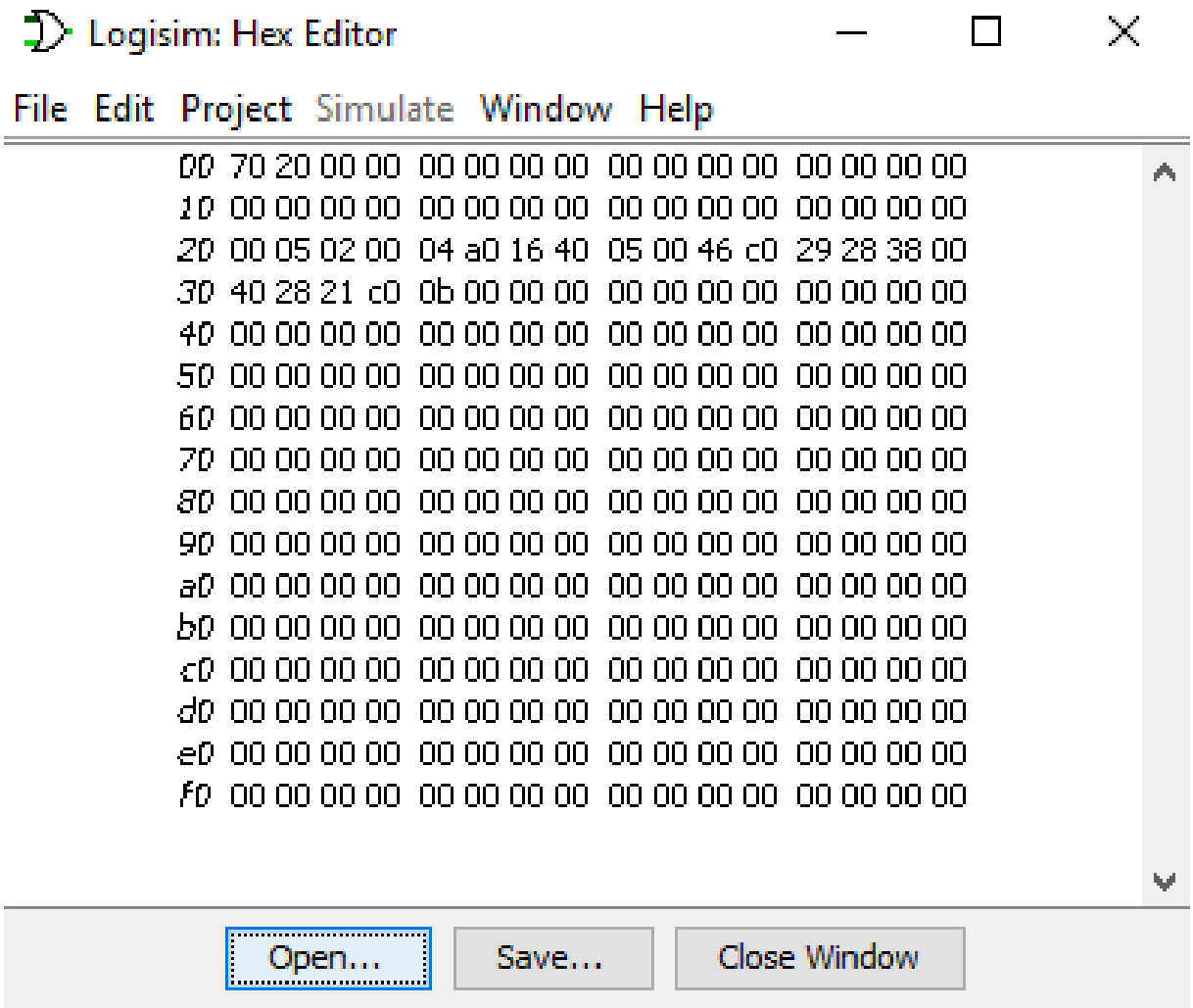

Figure 4: Full Circuit

We tried to test our implementation with the given example, but our circuit did not work properly. It did not stop at the desired T clock signal.

5 DISCUSSION

First of all, we designed a way to implement all of the opcodes which are given in pdf file by deciding whenever the selected bits are '1' according to time signals. Secondly, we sorted this list according to input variables. Both of the lists are added to the report.

6 CONCLUSION

In this project, we designed a basic computer. The project, this time around, was extremely long and tedious. We couldn't managed to track the mistakes we made on time. We implemented the circuit to the best of our abilities.
To conclude, this project helped us learn how CPU handles different kinds of instructions.